grepo - Repository group support

Srini, rs@toprllc.com

May 27, 2020

1 Introduction

Projects with many components and developers are best managed with distinct independent source code repositories. At a high level, there are the public repositories developed outside the team which are just used; the components that are unique to the project then may be viewed as private.

The public repositories then are primarily replicated for usage, prepared (eg. built) and periodically refreshed. The private repositories on the other hand go through an active development cycle. In git parlance, this would include creating new branches, merging, tagging and of course publishing. Specific sub projects thus may well span several repositories.

This tool enables developers manage their **projectlets** as a group. For example create a new **feature branch** on a set of related repositories and finally push or publish the branch as a group.

`https://source.android.com/setup/develop/repo` is such a tool which serves as a pattern for this projectlet. The goals are simplified somewhat:

- **git** is the only source code control system supported.
- Publishing updates, tags etc are to the original source repositories. The complete link to the repositories and the histories is maintained.
- Commands mirror their **git** equivalents.
- Project configuration is provided in **yaml** format.

1.1 Typical use cases

Developers of embedded systems using **yocto** or **embos** have to manage their environment as outlined above. Their own application might be split into several repositories while depending heavily on the platform libraries spread over numerous other repositories. Both these groups are evolving - the public ones presumably changing slower than the private ones.

2 Specifications

2.1 Configuration

A developer creates a Project configuration file eg. project.yaml listing all the public and private repositories. Uses the grepo tool to setup the initial checkout of the development base. For each feature, a feature branch is created in all the repositories. Development may affect some of the repositories while other repositories may not be affected at all.

Operations such as tagging, publishing all are performed in one transaction.

Publishing of the feature branch to the origin is predicated on actual changes. Repos which did not have any changes will not be published to the origin.

Listing 1: Example Project.yaml

```
public:
    workarea: ".worklib"
    reference: "V01.00"
    projects:
      - repourl: "git@gitlab.com:projtemplates/go.git"
        reference: "master"
        path: "go/lib"
        build: "make setup all"
private:
    workarea: ".work"
    server: "git@gitlab.com:"
    reference: "tagx02"
    projects:
      - repo: "RajaSrinivasan/icm.git"
        path: "icm"
        reference: "V01.06"
      - repo: "RajaSrinivasan/codex.git"
        path: "codex"
        reference: "v0.1.0-B"
        build: "make setup all test"
      - repourl: "https://github.com/RajaSrinivasan/srctrace.git"
        path: "tools"
      - repourl: "git@github.com:repotrace.git"
        path: "tools"
```

Project Groups The sample configuration lists two project groups: public and private. **public** projects are considered reference only and thus cloned and then detached from the HEAD. The repositories are considered immutable by this tool and thus push, tag and other operations will not be applied to the public projects.

Workarea Each project group requires a work area to be provided. This is the top level of the directory structure where the projects will be cloned. Each project's path specification is appended to the work area to determine the workarea for the project. For the project codex in the above example, the work area will be `.work/codex`. Please note that the path for each project has to be unique.

Server The project group can have a default server specified. This will be used for each project unless overridden in the project.

Repository spec A project can provide one of **repo** which is combined with the Server of the project group. Alternatively a **repourl** can be specified which becomes the full specification of the repository. In the above example, for **srctrace** the entire url is provided whereas for **codex** the url is formed using the Server spec and the repo spec resulting in: **git@gitlab.com:RajaSrinivasan/codex.git**.

Reference For default reference of the project group will be applied to each project unless overridden in the project. A reference could be a branch name or a tag. For the project, the reference could also be a commit id.

Build instruction A project can provide a build instruction. Upon completion of a clone or a pull, this command expected to be a shell command is executed. As an illustration, for the project codex in the example, a make command is specified.

3 Usage and examples

3.1 usage

```
        grepo supports a project that comprises different repositories.

Usage:
  grepo [command]

Available Commands:
  diff        Diff from where we started
  help        Help about any command
  init        Initialize - setup the workspace
  pull        Pull for each repo
  push        Push for each repo
  status      Project Status
  tag         Tag each repo
  version     Report the version of the application

Flags:
      --config string   config file. (default "Project.yaml")
  -h, --help            help for grepo
      --verbose         be verbose
  -v, --version         version for grepo

Use "grepo [command] --help" for more information about a command.
```

init The init command sets up the directory structure, clones the repository and performs the initial build of the repository.

pull After the initial setup, the pull command applies **git pull** to each of the projectlets. The public repos can be pull'ed optionally.

tag Applies the tag to each (private only) repository. While pushing, the tags will be pushed as well.

diff For each private repository, this displays the difference

push For each private repository, this performs a commit and a push. The same commit message is applied to all the repositories. Individual changes to the repositories must be "add"ed by the user. The repositories are all available for direct manipulation with native git.